**Assignment # 4**
**CSCI 201 Spring 2017**
**4.5% of course grade**

For this assignment, we will be taking a break from Cinemate and instead be implementing a stand alone application interfaced through the console. In order for you to gain as much experience as possible with networking and multithreading through this assignment, it is best if we deviate from web development. Web frameworks significantly help in handling networking and thread safety, which is why they are so widely used. For our purposes however, we want to give you a deep rooted understanding of server/client interaction and multithreading, which is why you will instead be implementing a Java application.

You will be implementing a slightly altered version of the game of hangman, which we will call Cineman (Like cinnamon? Get it?). In case you are unfamiliar with hangman, it is a simple guessing game generally played with two players using paper and pencil. Essentially, one player chooses a word or phrase and the other player tries to guess it. You can find a detailed description of the game here: http://www.wikihow.com/Play-Hangman (if you do not know how hangman is played, I highly recommend reading this).

In our version of hangman, the server will generate the word or phrase for the game, rather than a player getting to choose. The server will randomly choose the word or phrase from the actors, writers, and movie titles parsed from the xml file used in Cinemate, Assignment 3 (yay! An opportunity for software reuse!). You are guaranteed there will be no errors in the xml files used for testing.

Game Summary

There can be between 1-4 players that join the game. Each round, instead of the players taking turns in guessing, players can guess a letter at their convenience. If two players try to guess at the same time, the first one to reach the server gets to complete the turn. If the character exists in the phrase, the occurrences of that character should now be revealed on the console of all players (see Figure 9). If the character does not exist in the phrase, only the player who guessed it should have a limb added to their hangman. There are a few ways to win the game: a player guesses the entire phrase, a player wins by default when all other players complete their hangman (and are subsequently eliminated), or when a player guesses the last missing character in the phrase. It is impossible for no player to win and and impossible for more than one player to win.

Game Set Up

In order to start the game server, the server program should query for the port to host on. If the user provides a string, or a port that throws an IOException, an error message should print saying that the port was invalid. They should be queried for the port until a valid one has been provided. Next, the user must enter either the relative or absolute path of the xml file being used to generate the phrases for the games. Once the server has successfully been started and an xml file has been provided, print a message conveying this.

```
Please enter the port to host the server
port
Invalid port. Please enter the port to host the server
6789
Please enter the path to the xml file used for the game phrases
valid.xml
Server started!
```

Figure 1: Starting the server

If an instance of the client is started, there should be a welcome message. They should then be queried for the IPAddress and port to connect to the server. Once they provide valid credentials (you must handle the case when a string is provided for the port value, and when the port and IPAddress are unable to connect to the server), a message should print that they successfully connected to the server.

```
Welcome to Cineman!

Please enter the ipaddress
localhost
Please enter the port
port
Invalid port. Please enter the port
6666
Unable to connect to server with provided fields
Please enter the ipaddress
localhost
Please enter the port
6789
Congratulations! You have connected to the Cineman server!
```

Figure 2: Starting the client - examples of invalid inputs.

At this point, a menu should print with two options: the player can either start a game or join a game. If they enter an invalid command (a string, or an integer other than 1 or 2), an error message should print, and the menu should print again.

```
Please choose from the following:
1. Start a game
2. Join a game
nothing
Invalid command. Please choose from the following:
1. Start a game
2. Join a game
```

Figure 3: Client menu - example of invalid input

If they enter the command to start a game, they need to provide some information. First they should be queried for the username they wish to use. Next, they need to provide a name for their game. This will need to be unique so that the server can use it as the identifier for their specific game. If they try to use a name that is already being used for another *ongoing* game, a message should print, and they should be queried for a different name. Once they choose a valid game name, they should be queried for the number of players in the game. If they enter a string, a number less than 1, or a number greater than 4, an error message should print and they should be queried again for the number of players. If they enter 1 for the number of players, the game should start.

```
1. Start a game
2. Join a game
1
Please enter your username
emma
Please enter a unqiue name for your game
emma's game
this name has already been chosen by another game. Please enter a unique name for your game
howard's game
Please enter the number of players (1 - 4) in this game
nothing
Invalid number of players. Please enter the number of players (1 - 4) in this game
```

Figure 4: Inputs to start a game

If they choose more than 1 player, a message should print with how many players are left that need to join before the game can start. Each time a player joins, this message should print again with the username of the player that just joined and the updated number of players needed. Once the expected number of players have joined, the game can start.

```
Please enter the number of players (1 - 4) in this game
3
Waiting for 2 players to join..
matteo joined the game. Waiting for 1 players to join..
dana joined the game.
```

Figure 5: Starting a game with more than one player

If the user chooses to join a game, they first need to enter the name of the game they wish to join. If that the game doesn't exist or has reached the maximum number of allowed players, a message should print querying them for another game name.
If they successfully join a game, a congratulatory message should print.

```
Please choose from the following:
1. Start a game
2. Join a game
2
Please enter the name of the game you wish to join
dana's game
This game does not exist or has already reached the maximum number of players.
Please enter the name of the game you wish to join
howard's game
Congratulations! You have joined howard's game
```

Figure 6: Joining a game - invalid game name example

Once they have joined a game, they need to choose a username. If they try to choose a username that has already been chosen by another player in the game, a message should print alerting them of this, and they should be queried again for a username (once they provide a valid username, the host of the game should be notified that this player has joined).

Once they join the game, if more players need to join before the game can start, they should see a message saying "waiting for other players to join..". Once the required number of players has been met, they should see a list of the participating players before the game begins.

```
Please enter your username
emma
this username has already been chosen by one of the other players. Please enter your username
dana
waiting for other players to join..
all players have joined: dana, emma, matteo
```

Figure 7: Choosing a username and waiting for players - invalid username example

Game Play

When the game first starts, all players will see a printout of the UI. It should then print " <username>, please enter a character or the entire phrase as a guess. "

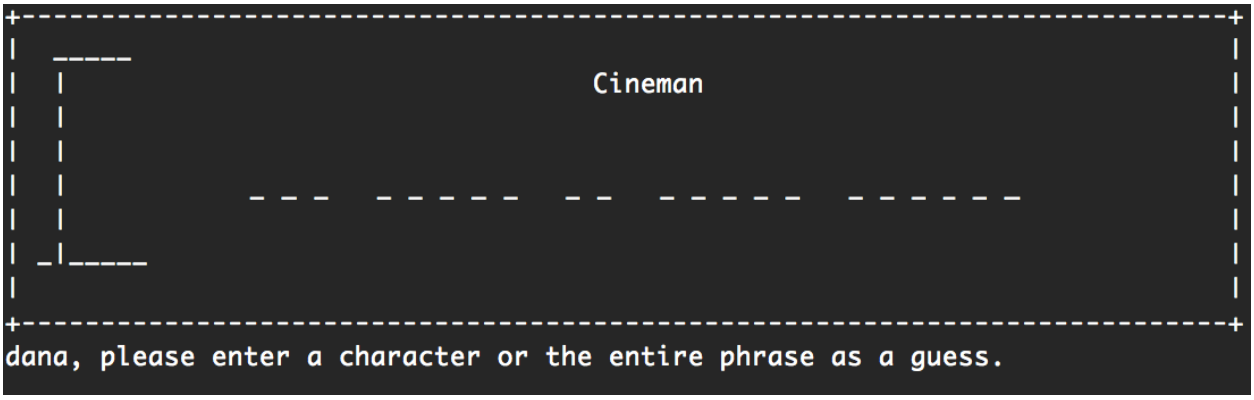


Figure 8 : From the perspective of dana (and matteo and emma ) - beginning of the game

If the current player enters a character, but another player guesses first, all players should see a message stating which player guessed what character. That character should either be listed at the bottom of the UI (see Figure 11) to indicate it was an incorrect guess, or the phrase should display every instance of the guessed character in it.

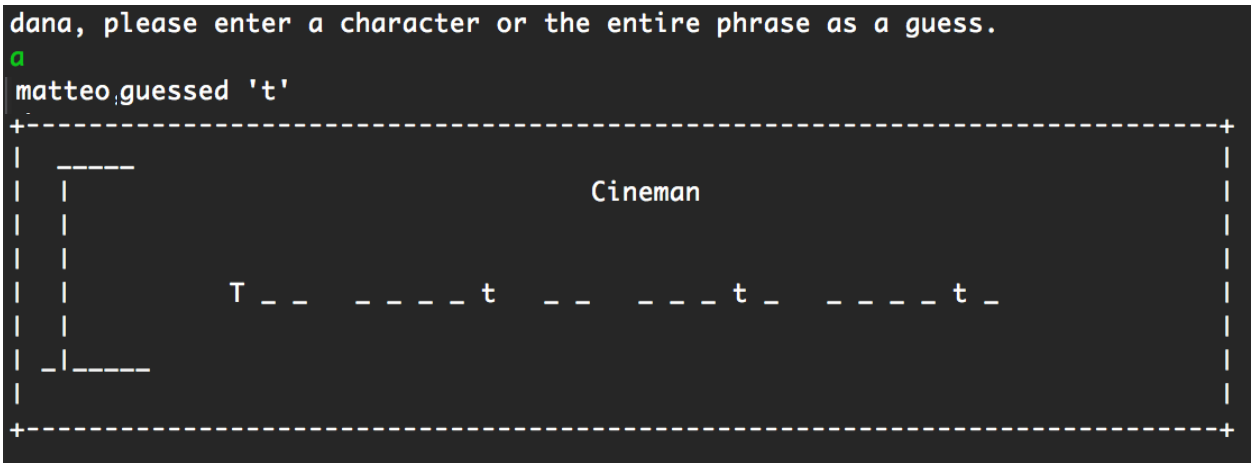


Figure 9: From the perspective of dana (and emma) - dana does not guess in time; matteo guesses a correct character

If the current player guesses the character, they should see a personalized message telling them whether their guess was correct or not.

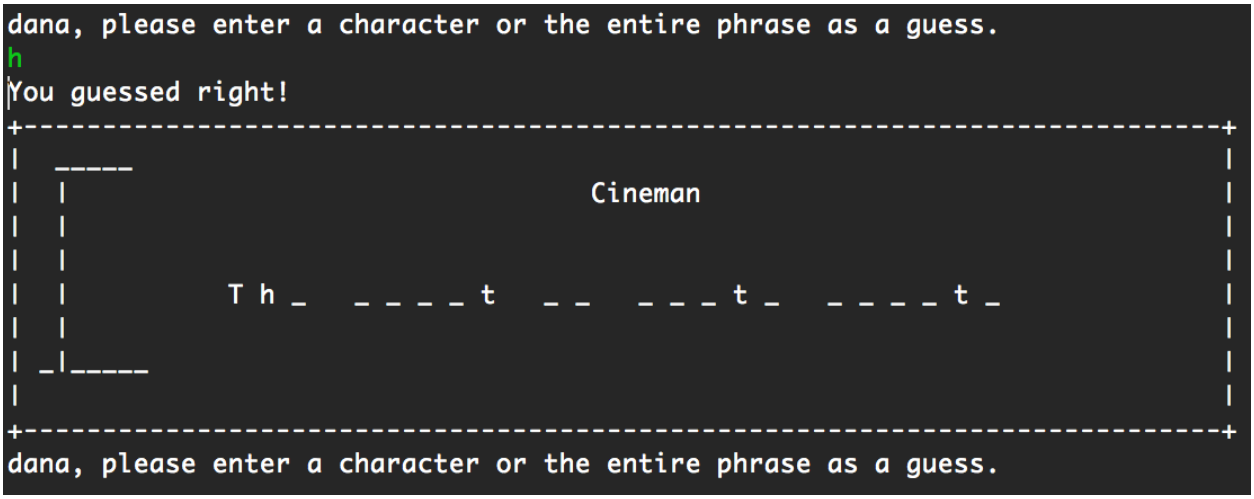


Figure 10: From the perspective of dana - dana correctly guesses 'h', and the updated UI is printed.

If the current player guesses incorrectly, they should see a sympathetic message indicating their guess was wrong. The updated UI should add a limb to the hangman (displayed on the left). The order in which the hangman should be constructed is head, torso, left arm, right arm, left leg, right leg, and finally, the noose connecting the head to the gallows. (see Figure ? for the final hangman image)

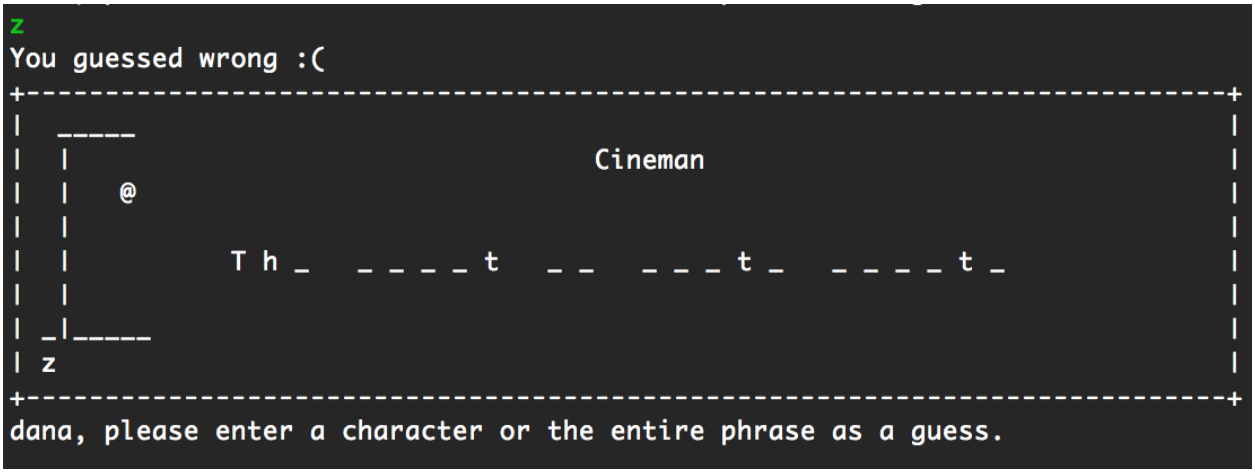


Figure 11: From the perspective of dana - dana guesses incorrectly. The character she guessed is added to the list at the bottom of the UI, and the head is added to her hangman

All other players should see an update stating that the player guessed incorrectly. Their list of chosen words should show the incorrect character that was guessed. However, a limb should **not be** added to the other players' hangmen. Only the player who guessed the incorrect character should have a limb added to their hangman.

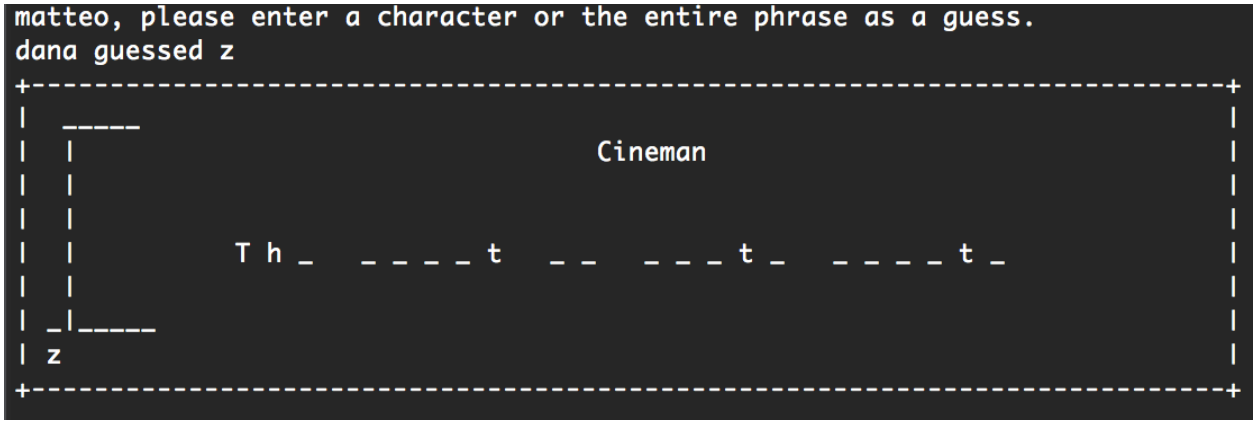


Figure 12: From the perspective of matteo (and emma) - dana guesses incorrectly; the list of (incorrect) character guessed now shows dana's guess; matteo's hangman is **not** added to

The players can try to guess the entire phrase as well (you can assume the phrase will always be two or more characters). If they guess incorrectly, a limb should be added to their hangman. The player who made the guess should see a message that their guess was incorrect.

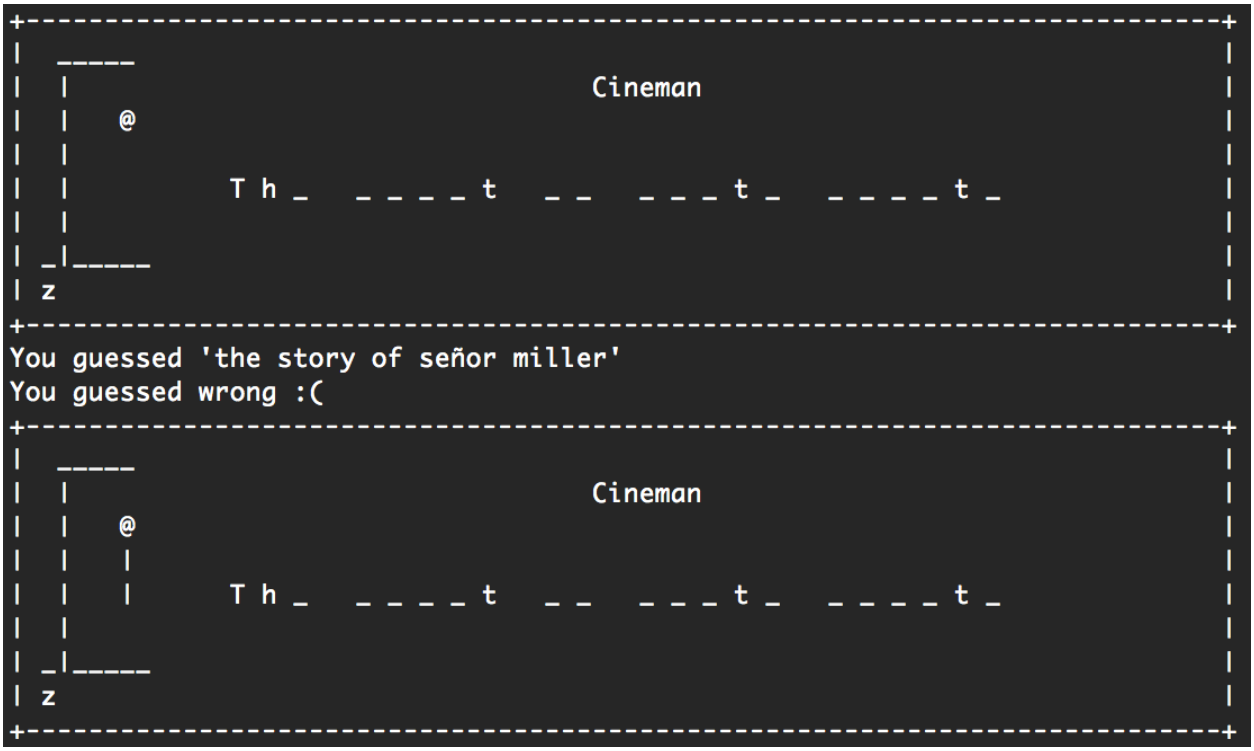


Figure 13: From the perspective of dana - dana incorrectly guesses the phrase. She sees a message that her guess was wrong and her hangman has another limb (the torso) added to it

All other players should see a message stating what phrase the player guessed.

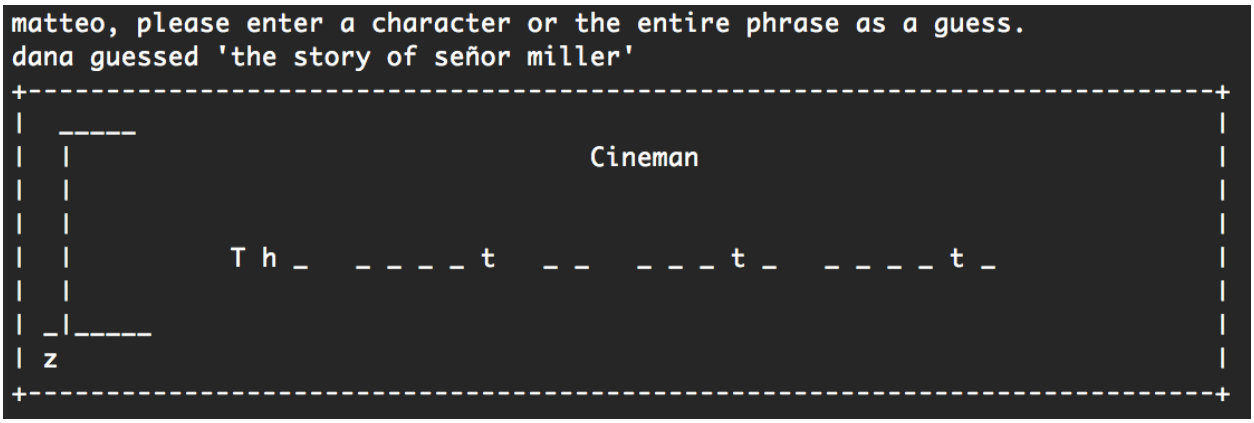


Figure 14 : From the perspective of matteo (and emma) - a message is printed with the phrase that dana guessed, and of course, the hangman is **not** added to

If a player guesses incorrectly too many times to the point where their hangman is completed, they should be eliminated from game play. Updates to the UI and messages should still print to their console as if they were still a part of the game. However, if they try to input a guess, they will receive a message that they can only spectate until the end of the game.

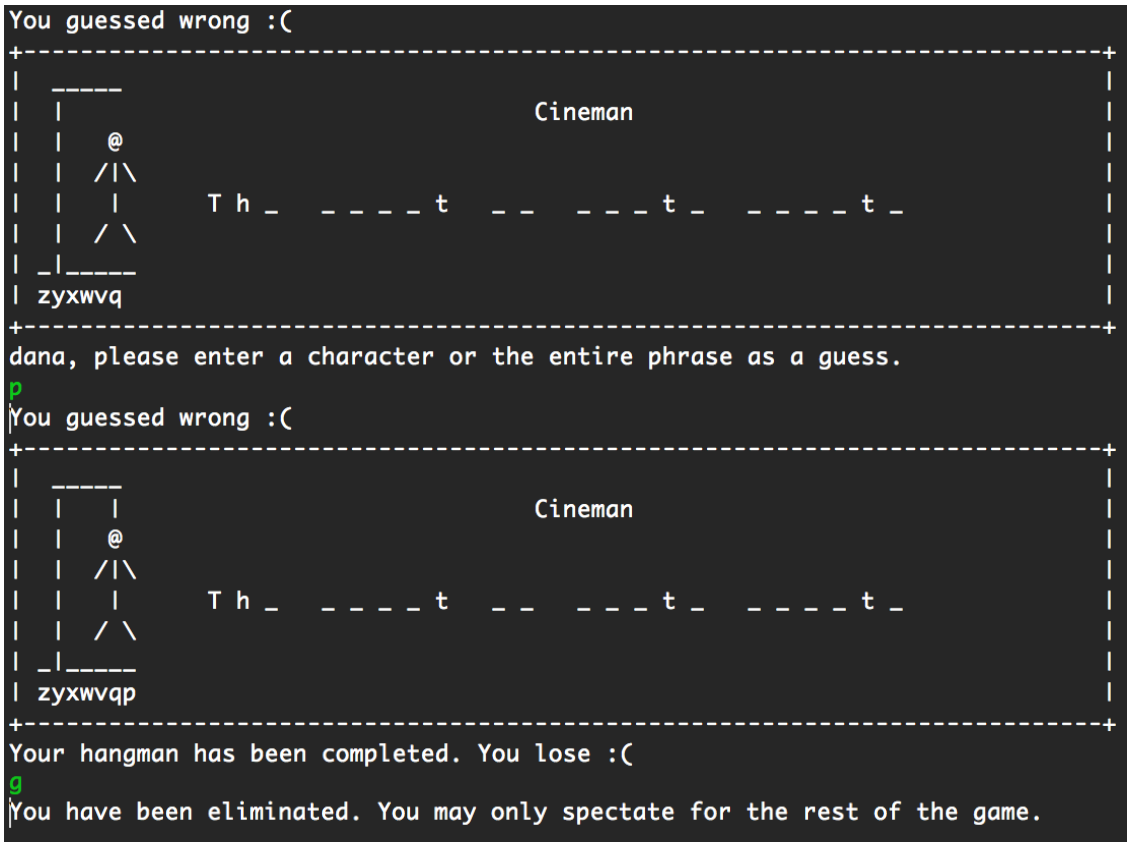


Figure 15: From the perspective of dana - dana guesses incorrectly and is eliminated. She tries to enter another guess but is informed she can only spectate

All other players should see a message informing them of when a player has been eliminated.

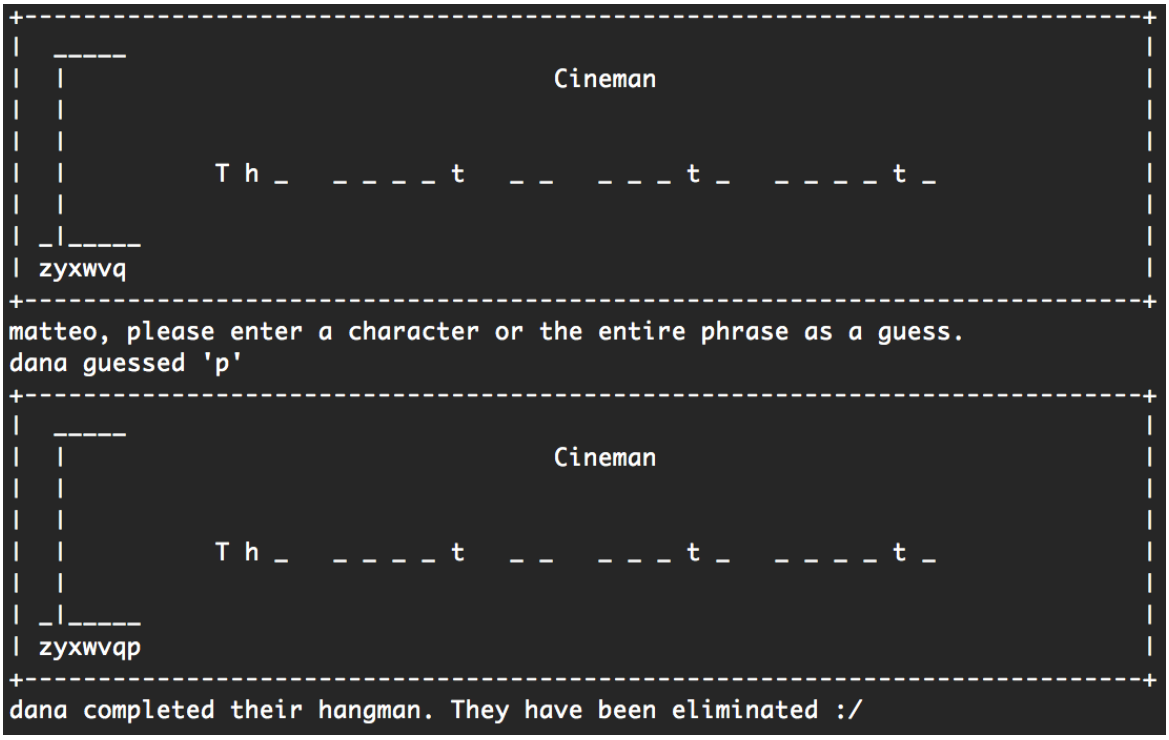


Figure 16: From the perspective of matteo (and emma) - dana is eliminated from game play after her guess; matteo receives a message that dana is no longer in play

The game continues this way until a player wins. When a player wins, all players should see a message with who won and who lost, and a 'game over' message. At this point, all the client programs should terminate, but the game server should still be running (and clients should still be able to connect to it and start another game if they wish).

Winning - Guessing the Phrase

One of the ways a player can win is by correctly guessing the phrase. If they do this, the UI should display all the characters filled in on the phrase (on all the players' consoles).


Figure 17: From the perspective of dana - dana guesses the phrase correctly

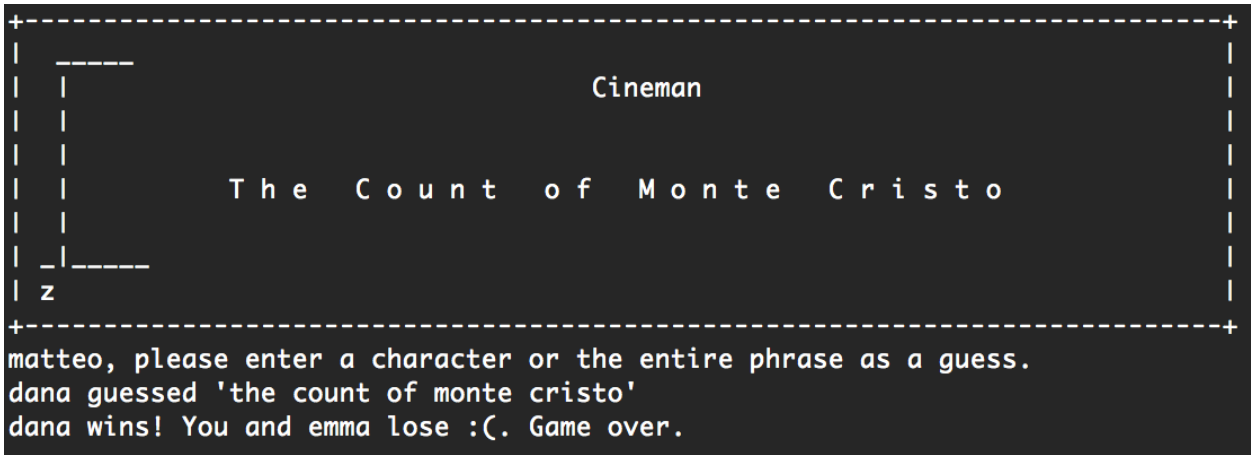


Figure 18: From the perspective of matteo (and emma) - dana guesses the phrase correctly; matteo sees a message that he and emma lost, and the game is over.

Winning - Guessing the Last Character Correctly

Another way in which a player could win is if they guess the last missing character of the phrase. Similar to guessing the phrase, the player who guessed correctly have a personalized message telling them they won.

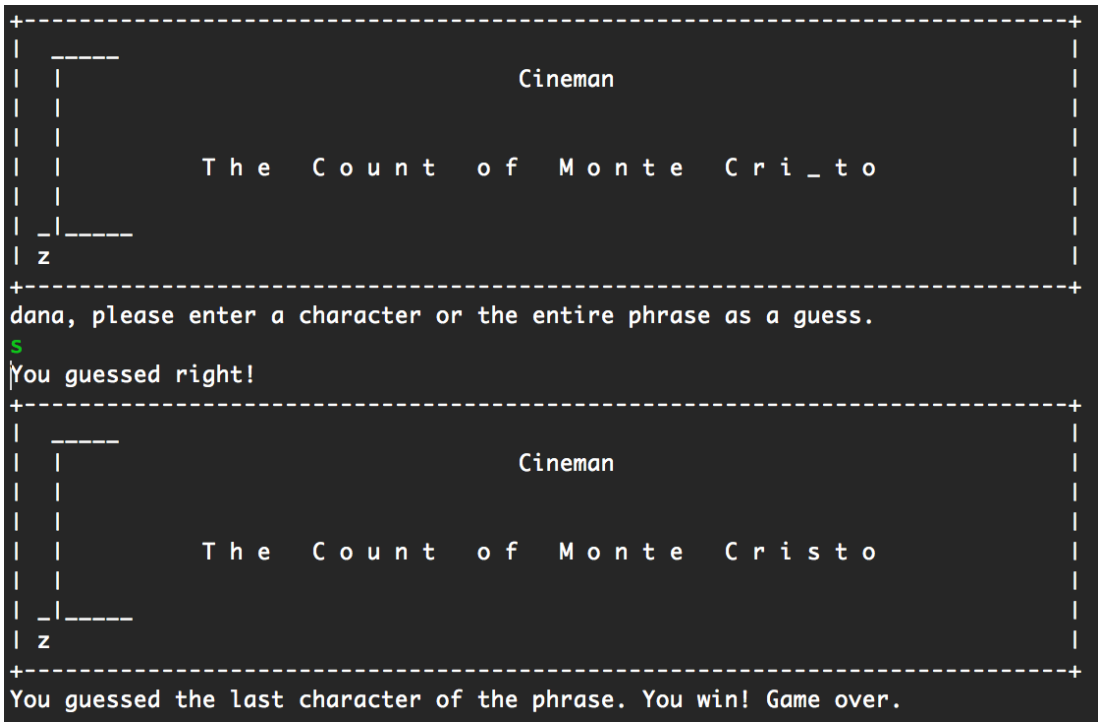


Figure 19: From the perspective of dana - dana correctly guesses the last character of the phrase and wins the game

All other players should see a message with who lost and who won, and that the game is over.

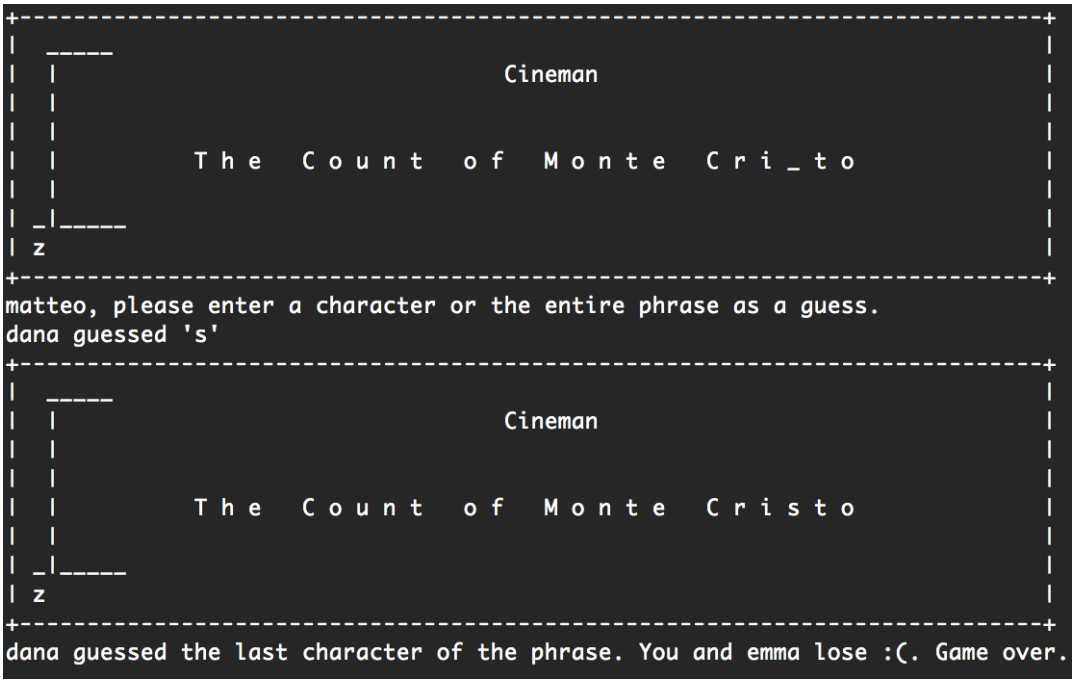


Figure 20: From the perspective of matteo (and emma) - matteo sees a message that he and emma lose, and that dana guessed the last character of the phrase, winning the game

Winning - All Other Players Eliminated (Winning By Default)

Lastly, a player could win by default. In other words, all other players were eliminated by completing their hangman, so the last player wins since they are the only player that has not been eliminated.

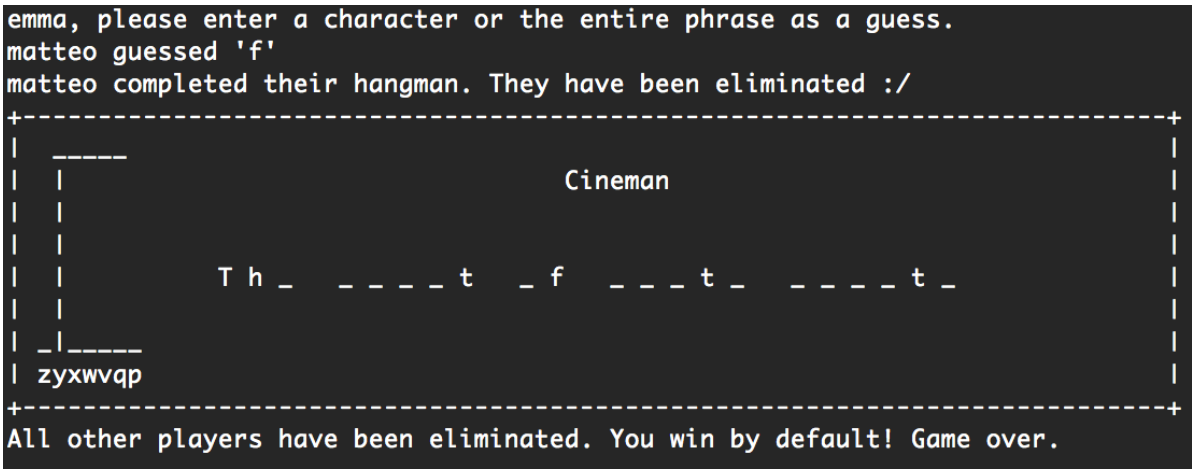


Figure 21: From the perspective of emma - emma sees the message that matteo has been eliminated (and it assumed dana has been eliminated as well). At this point, she sees a message that she won by default. Other players will receive a message telling them emma is the winner.

Other Notes

For consistency's sake, please try to print the exact same messages you see in the screenshot examples (rather than taking liberties with the wording).

However, the UI should look similar to the screenshots, although the border around it is not required. Your UI should have the hangman on the left , the title of the game (Cineman) at the top, the phrase in the middle, and the list of (incorrectly) guessed characters at the bottom. The grading on the UI will not be especially strict as we want you to be spending the majority of your time on the multithreading and networking.

**Grading Criteria (4.5%)**

Error Checking In Game Set Up (0.5%)

the program prints error messages and continues querying until a valid input is given if :

**Server:**

0.05% an invalid port is provided to the server

0.05% an IOException occurs during the server set up

**Client:**

0.05% an invalid port is provided to the client

0.05% the combination of the port and IPAddress provided to the client are unable to connect to the server

0.05% an invalid command is provided to the client menu

**Client - Start Game:**

0.05% the user tries to use a game name that has already been taken

0.05% the user enters a string value or a number not between 1 and 4 for the number of players

**Client - Join Game:**

0.05% the user tries to provide a game name that has already reached the player limit

0.05% the user tries to provide a game name that does not exist

0.05% the user tries to choose a username that has already been chosen by another player in their game

Game Set Up (1.0%)

0.1% The client is able to connect to the server when the correct IPAddress and port are provided

0.2% The server properly detects when a user tries to use an already existing game name when starting a new game

0.2% The server properly detects when a new player to a game tries to choose a username already chosen by another player in that game

0.3% The server properly detects when a new player enters the game and it displays a message to the game host with the number of players still needed and the username of the new player

0.2% The server properly prints the players in the game to players (except the host) before the game begins

Game Play (2.5%)

- 0.2% If a player correctly guesses a character, all players are notified and all instances of that character are now displayed in the phrase on their UIs
- 0.2% If a player incorrectly guesses a character, all players are notified and the character is added to the list of incorrectly guessed letters on their UIs
- 0.2% If a player incorrectly guesses a character only their hangman has a limb added to it
- 0.3% If a player correctly guesses the phrase, all players are notified who won and who lost, the phrase is revealed on the UI, and the game ends
- 0.2% If a player incorrectly guesses the phrase, all players are notified as to who guessed and what their guess for the phrase was. The player who guessed the phrase has a limb added to their hangman (and only that player)
- 0.3% If two players guess at the same time, only one of the guesses goes through to the server (first come first serve)
- 0.3% If a player makes an incorrect guess and their hangman is then completed, they see a message that they have lost and all other players are notified that this player was eliminated
- 0.2% If an eliminated player attempts to give a guess, they see a message that they are not able to guess anymore
- 0.3% If all but one players have been eliminated from the game, the last player wins by default. All players are notified of this and the game ends.
- 0.3% When the game is over, all client programs are terminated but the server remains running. If a user was to try to create another game (specifically one with the same game name as the game that just terminated), they should be able to do so without problems.

UI (0.4%)

- 0.2% The UI displays the hangman on the left, the hidden phrase in the middle, the list of incorrectly guessed characters at the bottom and the title of the game at the top
- 0.1% The hangman depiction looks similar to the examples given
- 0.1% The hangman has the limbs added to it in the order specified in the instructions

File Parsing (0.1%)

- 0.1% The phrase for the game is randomly chosen by the server from the writers, actors (their full names), and movie titles found in the Cinemate (Assignment 3) xml file provided.